[0355] A destructor implementation has a one-to-one association relationship with a destructor descriptor that is the destructor descriptor for which the destructor implementation provides an implementation. A destructor implementation has zero-to-many association relationship with precondition constraints that are each an implementation of a constraint to restrict some environmental condition or state of the model implementation before the execution of the destructor implementation. These types of constraints might check for the installation of a security manager, the existence of a database connection, or some other environmental condition. These constraints might also check to see if a model implementation has entered into the correct state to allow the destruction. Constraints related to the parameter values are held by the parameter descriptors.

[0356] A destructor implementation includes a destroy( ) operation that is the method call to perform a destruction.

[0357] An implementation change event is fired whenever an attribute value is added, changed or removed from the destructor implementation.

[0358] A destruction event with the status of the preconditions is fired to all interested parties when the destroy( ) method is called. A second event is sent at the successful completion of the operations related to destruction. A third event is fired only if a failure occurs during the execution of the destruction.

[0359] A destructor instance of the present invention may be unnecessary for many applications. The destructor implementation performs the work associated with destruction. The destructor implementation can always be retrieved from an instance by going through the metamodel implementation for that instance. A destructor instance does provide support for activities that model current reflection packages available in many programming languages.

[0360] A destructor instance has a one-to-one association relationship with a destructor implementation that is the implementation for which the destructor instance is an instance

[0361] A model implementation of the present invention extends another implementation. The virtual model implementation is defined by a model descriptor. The model implementation object contains implementations for each feature accessed by a model accessor. As discussed above, this may be as many as one feature for each feature in the model descriptor, or it may be less if the model accessor uses some non-virtual features.

[0362] A model implementation has a one-to-one association relationship with a metamodel that is the metamodel for which the model implementation provides an implementation. There exists a one-to-one relationship for each descriptor in the metamodel to each child implementation in this model implementation. A model implementation has a one-to-one specialization relationship with a parent model implementation that is the parent model implementation that is configured to give implementations for all the features of the parent model descriptor. This model implementation delegates to its parent model implementation wherever it inherits its features from it parent and does not override those features. A model implementation has a zero-to-one association relationship with a description that provides details about the hint for the correct use of this implementation and details about the implementation. A description is often useful for human users and automated documentation. This description is different from the description on the metamodel object. That description provides details about how the model is designed and why; this description provides details on use of the implementation. A model implementation has a one-to-one association relationship with a version that provides details about the number of modifications that have been made to the model implementation. A model implementation has a one-to-many aggregation relationship with a constructor implementation that provides a mechanism for creating new instances of the model described by the model descriptor. Several different constructors may exist, each of which uses a different number and type of arguments. A model implementation has a one-to-one aggregation relationship with a destructor implementation that provides a mechanism to destroy an instance of the model being described. Depending on the persistence rules associated with an implementation, this may permanently destroy the instance from the persistence store, or it may simply remove the representation of that persisted object from memory. The implementer will document the level of destruction. A model implementation has a zero-to-many aggregation relationship with static attribute instances that are attribute instances are held by a model implementation and therefore do not require an instance of the model in order to be accessed. These instances allow the value to be set and retrieved where the attribute is shared among all instances of the model. A model implementation has a zero-to-many aggregation relationship with instance attribute implementations that are attribute implementations that require an instance of the model in order to be accessed. These implementations allow the value to be set and retrieved on one specific instance. A model implementation has a zero-to-many aggregation relationship with static operation implementations that are operation implementations that do not require an instance of the model in order to be executed. These implementations provide a mechanism to execute the operation. A model implementation has a zero-to-many aggregation relationship with instance operation implementations that are operation implementations that require an instance of the model in order to be executed. These implementations provide a mechanism to execute the operation. A model implementation has a zero-to-many association relationship with signal implementations that provide the mechanism to register an appropriate instance's interest in receiving notification when the model generates an event. Also provides the mechanism to remove interest in receiving notification for an event. The instance registering interest in receiving event notification must implement the appropriate interface to match the listener type described in the signal descriptor. Signal implementations held by the model implementation are generally all of or a subset of those signals held by each attribute and operation (since the attributes and operations directly generate events and the model does not).

[0363] To register interest in events fired by an instance of a model implementation, use the signal implementations. This type of use applies to all implementations.

[0364] An implementation change event is fired whenever an attribute value is added, changed or removed from the model implementation. These attribute events relate to the structure of the model implementation, not the values held by the attribute implementations.